Preserving Privacy in Collaborative Filtering through Distributed Aggregation of Offline Profiles

Reza Shokri, Pedram Pedarsani, George Theodorakopoulos, and Jean-Pierre Hubaux
Laboratory for Computer Communications and Applications, EPFL, Switzerland
firstname.lastname@epfl.ch

ABSTRACT

In recommender systems, usually, a central server needs to have access to users' profiles in order to generate useful recommendations. Having this access, however, undermines the users' privacy. The more information is revealed to the server on the user-item relations, the lower the users' privacy is. Yet, hiding part of the profiles to increase the privacy comes at the cost of recommendation accuracy or difficulty of implementing the method. In this paper, we propose a distributed mechanism for users to augment their profiles in a way that obfuscates the user-item connection to an untrusted server, with minimum loss on the accuracy of the recommender system. We rely on the central server to generate the recommendations. However, each user stores his profile offline, modifies it by partly merging it with the profile of similar users through direct contact with them, and only then periodically uploads his profile to the server. We propose a metric to measure privacy at the system level, using graph matching concepts. Applying our method to the Netflix prize dataset, we show the effectiveness of the algorithm in solving the tradeoff between privacy and accuracy in recommender systems in an applicable way.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: [Information Filtering]; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Algorithms, Design, Security

Keywords

Recommender Systems, Privacy-Accuracy Tradeoff

1. INTRODUCTION

Recommendation systems are widely used to help users, overwhelmed by the huge number of options available to them, to find items that they might like. The items can be of any type: books, movies, web pages, restaurants, sightseeing places, online news, and even lifestyles. By collecting information about users' preferences for different items, a recommender system creates users' *profiles*. The preferences of a user in the past can help the recommender system to predict other items that might also be of interest to the user in the future. Collaborative filtering (CF), as one of the main categories of recommender systems, relies on the similarities of users' tastes: if two users have similar preferences for certain items, each user probably enjoys the items of interest to the other. Thus, the more each user gives information about his interests, the more meaningful the recommendations will be. This is the basis of CF systems.

In order to run the process of recommending items to users, recommender servers need to have access to users' profiles. Therefore, the profiles are usually stored on repositories to which collaborative filtering algorithms can have access. In such systems, the users do not have technical measures to limit the amount of information on their profiles to the server, that might not be necessary for generating recommendations. In other words, users have to put their profiles online (i.e., on the server) and trust the server (and the service providers) to keep the users' profiles private. The information available to the server hurts the privacy of the users on two levels.

First, if a user's real identity is available to the server, the server can associate the user's profile, which contains his private information, to his real identity (e.g., Amazon knows the real identities and postal addresses of those who have purchased products and can link them with the profile of those users). This is an obvious privacy breach, considering that a user does not want the link between his real identity and his profile to be revealed, yet he wants to use the service. This threat becomes more obvious when users share their opinion about the locations they regularly visit, e.g., restaurants, libraries, coffee shops, or sport centers [2].

Second, even if the real identity of a user is not known to the server, it can try to de-anonymize the user's identity by correlating the information contained in the user's profile and some information obtained from other databases [13].

Obviously, hiding information from the server helps to thwart these threats. Nevertheless, users want to receive accurate recommendations. Hence, the tradeoff between privacy and accuracy appears. The more accurate information the server has about users' profiles, the more meaningful the server's recommendations are, but the lower the users' privacy will be.

RecSys'09, October 23–25, 2009, New York, New York, USA.

Moving from centralized approaches to distributed collaborative filtering algorithms solves the privacy issues to a great extent [4, 11]. Yet, distributed CF algorithms are not as accurate as their centralized versions that have complete information about users' profiles. Moreover, they are not as practical as the centralized CF systems. The users' cooperation is needed not only to protect their privacy but also to make the system run properly. Making use of sophisticated cryptographic tools [6] might be another approach to solving the problem. However, these approaches are usually not practical and therefore remain unused. The security of these systems depends on that of their key establishment and key management. Considering that the perfect implementation of these schemes is hard to achieve in practice, they are not well received. Therefore, finding a more practical solution that does not trade much system accuracy in order to gain privacy for the users is an unsolved problem and yet crucial for users of CF systems.

In this paper, we propose a method that provides a compromise for the tradeoff between privacy on the one hand and accuracy and practicality on the other hand. In other words, our method increases the privacy in a practical way with a negligible effect on the recommendation accuracy. To this end, we still rely on the central server to generate recommendations. Additionally, we employ a distributed communication between users in order to improve their privacy. In our model, each user has two versions of his profile: the *online* (on the server) and the *offline* (located on the user's side), where the online profile is frequently synchronized with the offline version. Basically, the actual profile of a user is a subset of his offline profile. Users' offline profiles are aggregated in order to obfuscate the actual items rated by each user. As we will show, the hybrid nature of our approach helps users to gain privacy from distributed aggregation and to reach a level of accuracy comparable to the one achieved with a centralized CF.

The remainder of this paper is organized as follows. In Section 2, we define our notations and basic elements of a CF and state the problem, i.e., *privacy preserving in the presence of an untrusted server*. In Section 3, we provide an overview of the solution and elaborate our approach. In Section 4, we define evaluation metrics for privacy and recommendation accuracy. In Section 5, we validate the efficiency of our method by applying it to the real data set from the Netflix prize competition. Finally, in Sections 6 and 7, we review the related work and conclude the paper.

2. PROBLEM STATEMENT

2.1 Definitions and Notations

Formally, a collaborative filtering (CF) algorithm deals with a set of users and a set of items. In this paper, the non-empty set of users in the system is denoted by $\mathcal{U}$, where $|\mathcal{U}| = N$. We also represent the non-empty set of items by $\mathcal{I}$, where $|\mathcal{I}| = M$. Let $r_{u,i}(t)$ be the rating of user u to item i at time t, where $r_{u,i}(t) \in \{1, 2, \cdots r_{max}\}$ and r_{max} is the maximum valid rating. The set of items rated by user u up to time t is denoted by $\mathcal{I}_u(t) \subseteq \mathcal{I}$. The *profile* of user u at time t is defined as $\{(i, r_{u,i}(t))$ s.t. $i \in \mathcal{I}_u(t)\}$ which is the set of items coupled with their ratings rated by the user until t. We denote by $f_i(t)$ the *rating frequency* of item i at time t which is the fraction of users that have rated item i up to time t (i.e., $f_i(t) = |\{u \in \mathcal{U}$ s.t. $i \in \mathcal{I}_u(t)\}|/N$). To simplify the presentation, we sometimes omit the time index if it is clear from the context.

Collaborative filtering algorithms attempt to make predictions on the ratings of a particular user by collecting taste information from other users. CF algorithms fall into two general classes: *model-based* and *memory-based* methods [5]. Model-based algorithms learn a probabilistic model from the underlying data using statistical techniques, then they use the model to make predictions. Memory-based methods find similar users or items in the dataset in order to predict the items that a user might like and recommend them to him. This approach is based on the assumption that similar users prefer similar items, or that the preferred items of a user are similar. Memory-based CF methods can be further divided into two groups: *user-based* and *item-based* [16]. User-based methods, which are the focus of this paper, are heuristics to predict a rating of a user to an item by combining the ratings of users who are most similar to the target user (so called, the user's *neighbors*). The Pearson's correlation coefficient, as a popular measure [8], estimates the similarity $w_{u,v}$ between two users u and v, as follows.

$$w_{u,v} = \frac{\sum_{i \in (\mathcal{I}_u \cap \mathcal{I}_v)} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i \in (\mathcal{I}_u \cap \mathcal{I}_v)} (r_{u,i} - \overline{r}_u)^2 \cdot \sum_{i \in (\mathcal{I}_u \cap \mathcal{I}_v)} (r_{v,i} - \overline{r}_v)^2}} \quad (1)$$

where, $\overline{r}_u$ is the mean of ratings assigned by user u.

In order to predict the rating of user u to item i, first, the similarity of u to all other users is computed and then the nearest neighbors of u, denoted by set N_u, are determined. The set N_u contains $|N_u|$ users who are, based on (1), the most similar users to u. Next, the prediction of $r_{u,i}$, denoted by $\hat{r}_{u,i}$, is done by combining the ratings of neighbors of u to item i [8, 15], as follows.

$$\hat{r}_{u,i} = \overline{r}_u + \frac{\sum_{v \in N_u} w_{u,v}(r_{v,i} - \overline{r}_v)}{\sum_{v \in N_u} w_{u,v}} \quad (2)$$

Finally, the items that have a high potential of interest to the user (i.e., are predicted to have high and positive ratings) are recommended to him.

2.2 Problem Definition

We consider the scenario where a collaborative filtering algorithm (described in Section 2.1) is implemented on a server and users give information about their profiles to the server in order to receive recommendations. We define the problem as finding a mechanism by which the users can adapt their profiles (available to the server) to the privacy level they expect from the system. The solution must have minimum effect on the system's accuracy. We assume the server to be untrusted, and we evaluate the level of privacy in the system with respect to that. Yet, the information revealed among users themselves, if there is any need for communication, must be under the users' control (i.e., what information is given and to whom). Intuitively, the system privacy is high if the server is not able to construct the users' profiles based on the information available to it.

3. PROPOSED METHOD

In this section, we first give an overview of our solution in Section 3.1, and we define a few new concepts that we need to describe our scheme. Next, in Sections 3.2 and 3.3 we formally model the problem and our proposed solution.

3.1 Sketch of the Solution

We assume there is a central recommender system where users' profiles are stored and from which the users receive recommendations. We call a user's profile stored on the server his *online profile*. We assume that a user can have a local repository where he stores his own profile; we call these locally stored profiles the *offline profiles*. The server does not have access to the users' offline profiles and the recommendations are produced based on the online profiles available to the server. Each user independently synchronizes his online profile with his offline profile. Therefore, from time to time, the changes that have been made to the offline profile, since the last synchronization, will be applied to the online version, all at once. Obviously, the recently rated items are among these changes. In addition to these actual ratings, users add other items to their profiles, which are not originally rated by them; instead, they are received from other users with whom they communicate.

Users communicate with each other through different media such as face-to-face communication in a meeting, communicating over cellular networks, instant messaging through the Internet, communicating via a social network, or exchanging emails. We refer to any such communication as *contact* between two users. A user arbitrarily selects his peers and certain information about the users' offline profiles is exchanged between them at each contact. Based on this information, they add a subset of the other user's items to their own profiles. The exchanged information between users, the number of items that are added to their profile and the items (plus their associated ratings) that are selected for the aggregation depend on the *aggregation* function they use.

From the server's point of view, each user adds a batch of new items (plus the ratings) to his online profile at each synchronization. It is not known to the server which of these items have been actually rated by the user. In other words, the user's actual profile is hidden from the server; hence, there is privacy for the user. This, of course, comes at a cost: a drop in the recommendation accuracy. In the next sections, we answer the following questions, in order to find the appropriate aggregation functions: How many items should be aggregated in each contact? What items are better candidates for aggregation? What is the effect of contact rate on the privacy and accuracy?

3.2 The Model

We model users' profiles by a bipartite graph, where nodes denote the users and items, and weighted edges correspond to the ratings of the users to the items.

Let $G_t(U, V, E)$ be a bipartite weighted graph, where the vertex set $U = \mathcal{U}$ corresponds to the users, the vertex set $V = \mathcal{I}$ corresponds to the items, and the edge set $E(t)$ corresponds to the users' rating for the items until time t. Thus, an edge $(u, i) \in E(t)$ exists when user $u \in U$ has rated item $i \in V$ at some time before t. Each edge (u, i) has an associated weight $r_{u,i}(t)$ denoting the rating of user u assigned to item i. In such a setting, a user's profile or $\mathcal{I}_u(t)$ will be the set of nodes adjacent to node u. We call G_t the *actual* graph, because it represents the actual users' ratings of the items, i.e., in the case that there is no privacy preserving method in use.

Using our method, there will be two other graphs in the system. One should represent the users' online profiles and the other should model the users' offline profiles.

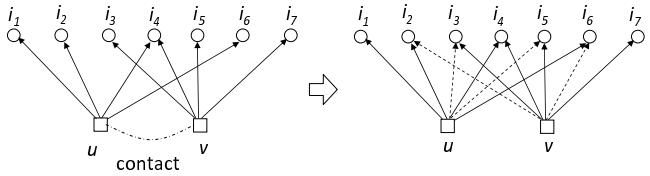


Figure 1: Aggregation of users' offline profiles after a contact. A contact occurs between users u and v at time t, with $\mathcal{I}_u^{\text{off}}(t) = \{i_1, i_2, i_4, i_6\}$ and $\mathcal{I}_v^{\text{off}}(t) = \{i_3, i_4, i_5, i_7\}$. User u gives $\{i_2, i_4, i_6\}$, which is a subset of his profile, to v. User v also gives $\{i_3, i_5\}$ to u. Additional edges (dashed lines) appear after the aggregation, and the offline graph is evolved through aggregating the new added edges. After the aggregation, $\mathcal{I}_u^{\text{off}}(t^+) = \{i_1, i_2, i_3, i_4, i_5, i_6\}$ and $\mathcal{I}_u^{\text{off}}(t^+) = \{i_2, i_3, i_4, i_5, i_6, i_7\}$.

Let us denote the *offline* graph by $G_t^{\text{off}}(U, V, E^{\text{off}})$. This conceptual graph, which is stored in a distributed manner, is formed by the aggregation of edges through users' contacts and by rating new items over time until t. The edges that are added to the offline graph can appear either as a result of actual ratings by the users, or through aggregation by contacting other users. Hence, the actual graph G_t is embedded and hidden in G_t^{off}. We also denote $\mathcal{I}_u^{\text{off}}(t)$ to be the user's offline profile at t. By definition, for every user u at any time t we have $\mathcal{I}_u(t) \subseteq \mathcal{I}_u^{\text{off}}(t)$. Note that $\mathcal{I}_u^{\text{off}}(t) \setminus \mathcal{I}_u(t)$ is the set of items (plus their ratings) that are added through aggregation to a user's offline profile.

In our model, the *online* graph is denoted by $G_t^{\text{on}}(U, V, E^{\text{on}})$ and $\mathcal{I}_u^{\text{on}}(t)$ represents the online profile of user u. The online graph gets updated over time when users synchronize their online profiles with their offline versions. As a result, at any time, there might be some edges in graph G_t^{off} that have not yet been added to the graph G_t^{on}. Therefore, $\mathcal{I}_u^{\text{on}}(t) \subseteq \mathcal{I}_u^{\text{off}}(t)$, for every user u at any time t.

Figure 1 illustrates the effects of aggregation on the offline profiles of two users after their contact. Assume two users u and v contact each other at time t, with offline profiles $\mathcal{I}_u^{\text{off}}(t)$ and $\mathcal{I}_v^{\text{off}}(t)$ before the contact. We also denote $\mathcal{I}_u^{\text{off}}(t^+)$ and $\mathcal{I}_v^{\text{off}}(t^+)$ to be their offline profiles after contact. As it is shown, each user (e.g., u) gives a subset of his own profile to his peer (e.g., v) to be aggregated to the peer's previous profile. The way they select the mentioned subset is explained in the next section. Note that the added edges preserve their weights (ratings). In the case the receiver is given an item that is already in his offline profile, he updates its rating, if it is not an actual rating.

We assume each user, on average, contacts n users per time unit and the contact peers are selected arbitrarily from $\mathcal{U}$. The process of contacting other users and updating the offline profiles continues for all users over time, in parallel with the process of actually rating new items. Thus, graph G_t^{off} will be accumulated. More formally, for $t_1 < t_2$, $\mathcal{I}_u^{\text{off}}(t_1) \subseteq \mathcal{I}_u^{\text{off}}(t_2)$, i.e., the resulting bipartite graph $G_{t_2}^{\text{off}}$ has the same vertex sets as $G_{t_1}^{\text{off}}$, and $E^{\text{off}}(t_1) \subseteq E^{\text{off}}(t_2)$.

Each user occasionally synchronizes his offline and online profiles at arbitrary time instants. Thus, the online graph on the server changes dynamically over time. Figure 2 shows this evolution.

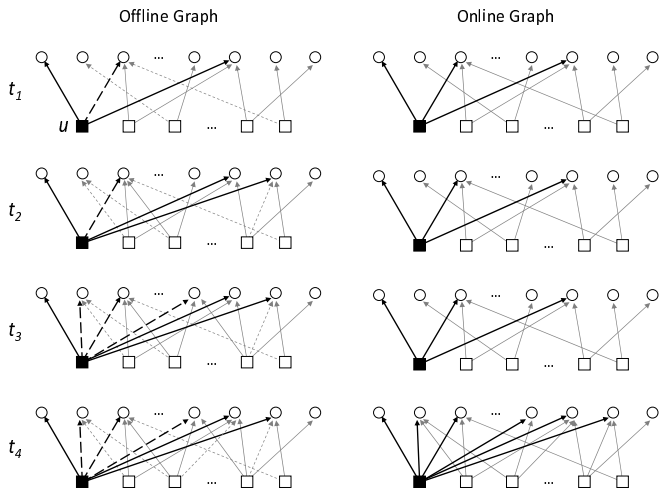


Figure 2: Evolution of the offline (left column) and online (right column) graphs over time ($t_1 < t_2 < t_3 < t_4$). Circles represent the items and the users are shown by the squares. We focus on the profiles of user u, distinguished by the black square. In the left column, the solid lines are the actual ratings of the users, and the dashed lines are the additional ratings aggregated to the users' offline profiles. Profiles $\mathcal{I}_u^{\text{off}}$ and $\mathcal{I}_u^{\text{on}}$ are synchronized at time t_1. Graph G_t^{on} stays unchanged, regarding user u, until the next synchronization time t_4. Graph G_t^{off} is accumulated through addition of both actual rating edges (solid lines) and also the aggregated edges (dashed lines). User u rates a new item at t_2 and aggregates some other items at t_3. The solid lines in the right column (i.e., the online graph) indicate that the server is unable to distinguish between the actual and dummy (i.e., those added through aggregation) ratings. The grey lines belong to other users' profile that are evolving independently of each other.

We emphasize that each time a user synchronizes his online profile with the offline version, the server is incapable of distinguishing between the new edges that belong to the user's actual profile and the edges added through aggregation. Moreover, users who contact each other have access only to information derived from the offline profiles of their peers and cannot pinpoint the actual items of each others' offline profiles. Hence, a user's privacy is protected not only against the server but also against the other users.

In order to prevent a user being confused about his own rated items, the process of aggregation and also the aggregated items (i.e., his offline profile minus his actual profile) can be made transparent to the user. Hence, a user can add, remove, or modify his actual ratings without any confusion, although he is able to find out what the extra items are in his profile. Looking from the server's side, the server is not aware of the items that are actually rated by the user, and generates recommendations based on their online profiles. Therefore, the server passes the top items that might be of interest to a user without filtering out the items already existing in his profile. Instead, the process of filtering is done at the user's side, in a transparent way, i.e., the server recommends a set of highly recommended items to the user and it is the user application that avoids recommending the items that are previously rated by the user himself. This prevents a user from missing a recommendable item because it is in his profile but not actually rated by the user.

3.3 Profile Aggregation

As described in the previous section, the process of updating graph G_t^{off} is accomplished through an aggregation process where users contact each other and update their offline profiles by adding a subset of their peer's rated items to their own profile. Consider a contact between users u and v at time t. We denote $\mathcal{I}_v^{\text{agg}}(t)$ (named the *aggregated items* from v) as the subset of items (plus their ratings) in the offline profile of user v, which are added to the offline profile of user u. Considering $\mathcal{I}_u^{\text{off}}(t^+)$ as the offline profile of u after the aggregation, the following equation holds:

$$\mathcal{I}_u^{\text{off}}(t^+) = \mathcal{I}_u^{\text{off}}(t) \cup \mathcal{I}_v^{\text{agg}}(t). \tag{3}$$

The same argument holds for the inverse case (updated profile of v). Note that the added edges keep the same rating value after being added to a user's profile, i.e., assuming item i is added to the offline profile of user u after his contact with user v, we have:

$$r_{u,i}^{\text{off}}(t^+) = r_{v,i}^{\text{off}}(t), \quad i \in \mathcal{I}_v^{\text{agg}}(t), i \notin \mathcal{I}_u(t). \tag{4}$$

But, if the candidate item already exists in u's actual profile, i.e., $i \in \mathcal{I}_v^{\text{agg}}(t) \cap \mathcal{I}_u(t)$, the rate stays unchanged.

In order to choose the candidate items for aggregation, we should consider two issues: 1) The number of items to be aggregated ($|\mathcal{I}_v^{\text{agg}}(t)|$), and 2) which subset of $\mathcal{I}_v^{\text{off}}(t)$ (with cardinality $|\mathcal{I}_v^{\text{agg}}(t)|$) to choose. We provide answers to these questions by introducing different types of aggregation.

In this work, we introduce two kinds of possible aggregation processes, focusing on our key idea: aggregation based on the similarity of the users who contact each other. We believe that this approach yields the best performance of the system for preserving both privacy and accuracy.

3.3.1 Similarity-Based Aggregation

In this case, at each contact, the similarity of the two users is calculated using (1). Each user then gives a proportion - equal to the similarity value - of his items in his offline profile to the other user for aggregation, i.e., if we assume a contact between users u and v at time t, using (3) we have,

$$|\mathcal{I}_u^{\text{off}}(t^+)| \leqslant |\mathcal{I}_u^{\text{off}}(t)| + \lfloor sim_{u,v} \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor, \tag{5}$$

where $|\mathcal{I}_u^{\text{off}}(t^+)|$ denotes the size of u's offline profile after aggregation, and $|\mathcal{I}_u^{\text{off}}(t)|$ and $|\mathcal{I}_v^{\text{off}}(t)|$ denote the size of u and v's profiles before aggregation, respectively. Note that $|\mathcal{I}_v^{\text{agg}}(t)| = \lfloor sim_{u,v} \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor$, and the equality holds when $\mathcal{I}_u^{\text{off}}(t) \cap \mathcal{I}_v^{\text{off}}(t) = \phi$. We denote $sim_{u,v}$ as the similarity between users u and v (value between 0 and 1) is computed as follows:

$$sim_{u,v} = \begin{cases} w_{u,v} & \text{if } w_{u,v} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In other words, the two users only accept the aggregated items when there is a positive similarity between them, otherwise no aggregation occurs.

Although our main focus is on the protection of the users' privacy against the untrusted server and not against each

other, we still can employ some tools to protect users' privacy from each other. Users, who contact each other, need to reveal their profiles to each other in order to compute the similarity value. The level of privacy can be improved by using methods that compute similarity between two profiles without revealing their content. Lathia *et al.* [9] propose an interesting concordance measure to estimate the similarity between two users in a distributed system without revealing their profiles to each other. This method can be used when one user contacts another user in whom he has little trust.

To answer the question of "which subset of $\mathcal{I}_v^{\text{off}}(t)$ to choose", we notice that the items in $\mathcal{I}_v^{\text{agg}}(t)$ can be chosen through different methods. We hereby suggest two ways:

- **Similarity-based Minimum Rating Frequency (SMRF)** One way to choose the set $\mathcal{I}_v^{\text{agg}}(t)$ is to sort the elements of $\mathcal{I}_v^{\text{off}}(t)$ by their rating frequency, and select the first $\lfloor sim_{u,v} \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor$ elements with minimum rating frequency. In other words, we choose the subset of the user's rated items that have been least rated by the users in the system (i.e., in the online graph). We assume that the server makes the current rating frequency of items, $f_i^{\text{on}}(t)$, available to the users.

- **Similarity-based Random Selection (SRS)** In this case, the $\lfloor sim_{u,v} \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor$ candidate items are selected uniformly at random from the set of items in $\mathcal{I}_v^{\text{off}}(t)$.

We evaluate the effectiveness of these aggregation functions and compare them in Section 5.

3.3.2 Fixed Random-Selection Aggregation

Aside from choosing the aggregated items based on the similarity of users, we consider another simple type of aggregation: *fixed random selection*, where each user gives a *fixed* proportion of his rated items to the other user, i.e.,

$$|\mathcal{I}_u^{\text{off}}(t^+)| \leqslant |\mathcal{I}_u^{\text{off}}(t)| + \lfloor k \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor, \tag{7}$$

where, k is a constant value between 0 and 1. Note that here $|\mathcal{I}_v^{\text{agg}}(t)| = \lfloor k \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor$, and the aggregated items are chosen uniformly at random.

- **Union** is an example of this aggregation, where each user gives all of his rated items to the other user, i.e.,

$$\mathcal{I}_u^{\text{off}}(t^+) = \mathcal{I}_u^{\text{off}}(t) \cup \mathcal{I}_v^{\text{off}}(t). \tag{8}$$

4. EVALUATION METRICS

In this section, we describe our definition of privacy and recommendation accuracy. Following the definitions, we formalize our metrics for evaluating the proposed method in terms of the privacy gain and the accuracy loss.

4.1 Privacy Measurement

We define the *lack of privacy* as the amount of information the server has about the actual profile of the users. In other words, it reflects how accurately the server can guess the actual profile of the users (modeled as G_t) using the online graph G_t^{on}, and how valuable the estimated profiles are, in terms of identifying the users and distinguishing between them. To define the privacy, we focus more on the users-items connection rather than on the ratings the users assign to the items (for an adversary who tracks a user it is more interesting to know to which places the user has been, rather than knowing the user's opinion about those places).

To evaluate the privacy provided by our method, based on the above-mentioned privacy definition, we define a privacy metric considering the graphs G_t and G_t^{on}. We emphasize that privacy is preserved in our model through additional edges that exist in the online graph, but not in the actual graph. Thus, we compute to what extent the structures of graphs G_t and G_t^{on} are similar, as a higher structural difference accounts for higher privacy. This falls into the subject of structural graph matching, where the structures of two graphs are compared with the goal of matching the correspondent nodes based on the structures. In our problem, as the node set of the two graphs are the same, the structural difference can be viewed as the difference between the correspondent edges. This idea has also been investigated in the field of pattern recognition where the *edge-consistency* of two graph patterns (in matching a data graph to a model graph) is used to obtain the correspondence errors [10, 12].

Following the above discussion, we introduce the error measure for edge-inconsistency, considering only the structures of two given graphs $G_1(V, E_1)$ and $G_2(V, E_2)$, where V denotes the node set (the same for both graphs), and E_1 and E_2 denote different edge sets. The matching error Δ can be generally defined as the maximum likelihood estimator for edge differences over the edge sets of G_1 and G_2, i.e., the number of edges that exist in one graph with their correspondent edges not existing in the other graph:

$$\Delta = \sum_{(u,i)\in E_1} \mathbf{1}_{\{(u,i)\notin E_2\}} + \sum_{(u,i)\in E_2} \mathbf{1}_{\{(u,i)\notin E_1\}}, \tag{9}$$

where $\mathbf{1}_{\{A\}}$ denotes the indicator function on A, and (u, i) denotes an edge between nodes u and i in either of the two graphs.

In our setting, graphs G_t and G_t^{on} correspond to G_1 and G_2 respectively, with their node set divided into two sets V and U to exhibit the bipartite property of users and items. The privacy metric is equivalent to the matching error Δ, as Δ counts the number of differences in the corresponding edges of two graphs, and higher structural difference results in higher privacy. In our case, the first summation in (9) is zero, because G_t^{on} is an accumulated version of G_t. Consider our own notations, where $\mathcal{I}_u^{\text{on}}$ and $\mathcal{I}_u$ stand for the items associated with user u in his online and actual graphs, respectively. Counting the edges by iterating over the user set U, the matching error can be written as:

$$\Delta = \sum_u \sum_{i\in\mathcal{I}_u^{\text{on}}} \mathbf{1}_{\{i\notin\mathcal{I}_u\}}. \tag{10}$$

However, Narayanan and Shmatikov [13] show that an item with a high rating frequency contains less information about those who have rated the item than an item with a low rating frequency (e.g., it is more valuable, in identifying a user, to know that a user has purchased "The Color of Pomegranates" than the fact that he purchased a Harry Potter DVD). Their result shows that the higher the rating frequency of an item is, the more important this item is for identifying users associated with that item, thus the more valuable it is in system privacy preservation. If an adversary (the one who wants to break users' privacy) falsely believes that someone has rated an item with a low rating frequency, then the user's privacy is much higher than the case where the item has a high rating frequency.

Hence, we use the rating frequency f_i in (10) to weight the items. We then normalize it per user by dividing it by its maximum value for each user. This also guarantees a value between 0 and 1 for the system privacy. Thus, privacy in the current work can be defined as the normalized weighted edge-difference function over the node pairs of the two graphs. Rewriting (10), this can be expressed as follows. For the sake of simplicity, we omit the time index.

$$privacy = \frac{1}{N} \cdot \sum_{u} \left(\frac{\sum_{i \in (\mathcal{I}_u^{\text{on}} \setminus \mathcal{I}_u)} \left(\frac{1}{f_i}\right)}{\sum_{i \in \mathcal{I}_u^{\text{on}}} \left(\frac{1}{f_i}\right)} \right) \qquad (11)$$

To put is simply, looking at the bipartite graph, for each user we compute the weighted difference of his rated items in graphs G_t and G_t^{on}, with the weights being the inverse of the item's rating frequency. The overall *system privacy* is then computed as the normalized sum of all such terms for all users. Such a metric captures both the server's chance for successfully guessing the users' actual profiles and the importance of the profiles in identifying the users (using the rating frequency).

Let us have a deeper look at how the privacy metric can be interpreted. The closer is the privacy degree of a system to 1, the harder is for the adversary to distinguish the actual profiles of the users in their online profiles. Moreover, it becomes harder for the adversary to distinguish between users and de-anonymize their profiles [13]. As it approaches 1, the privacy growth becomes slower by adding more items to the profile of users, whereas its growth is faster for small privacy values. The privacy becomes 1 if the actual graph is infinitely small compared to the online graph and it is 0 if they are the same. It is important to note that this metric is not designed to compare the privacy of two systems with different settings, rather to reflect the privacy gain inside a system.

Note that there are some items in a user's offline profile whose rating changes due to 1) being rated later by the user himself, 2) being received again from contacting users. The first subset is not distinguishable from the second and their ratio intuitively follows (11). Therefore, we do not re-evaluate the effect of this factor on the users' privacy.

4.2 Recommendation Accuracy

Modifying the users' actual profiles in order to increase their privacy level might impose some error on the accuracy of the recommendation system. We measure the cost of running our method as the difference between the recommendation error in our system and the system with no privacy. In both cases we compute the recommendation error based on RMSE. Let $\widehat{\mathcal{I}}_u$ denote the set of predicted items in the recommender system for user u. The system error for any graph G is computed as follows (the same for graph G^{on}).

$$RMSE(G) = \sqrt{\frac{\sum_u \sum_{i \in \widehat{\mathcal{I}}_u} (r_{u,i} - \hat{r}_{u,i})^2}{\sum_u |\widehat{\mathcal{I}}_u|}} \qquad (12)$$

Finally, the cost of using our method is computed as the percentage of the accuracy loss, i.e., the additionally imposed error, formalized as follows.

$$accuracy\ loss = \frac{RMSE(G^{\text{on}}) - RMSE(G)}{RMSE(G)} \qquad (13)$$

5. EXPERIMENTAL RESULTS

In this section, we validate the effectiveness of our model and the mechanisms we proposed in Section 3. First, we describe the data set that we used for the experiments and explain our simulation model. Next, we evaluate the effectiveness of different aggregation functions used in our mechanism based on the metrics described in Section 4.

In each experiment, we used a subset of the Netflix data set, released for Netflix prize [1], containing 300 randomly chosen profiles with ratings between early 2001 and late 2006. The ratings are on a scale from 1 to 5 (integral) stars.

To evaluate the recommendation accuracy, we used 10% of the actual ratings of each user as the testing set and the remaining 90% as the training set. For each experiment, we evaluated the privacy achievement and also the accuracy of the system for different values of users' contact rates. The contacts between users were selected uniformly at random, both for the contacting peers and their time of contact. We evaluated the privacy and accuracy of an aggregation function with respect to the average users' contact rate. We made the evaluation at the end of each experiment (December 2006). The contact rate value determines the average number of users a given user has contact with per *year*. Besides, in all experiments the number of neighbors of each user, needed for generating recommendations, was selected to be 30 (i.e., $|N_u| = 30$ for any user u). Outcome of various experiments are averaged to obtain the final results. In this setting, we implemented the following aggregation functions.

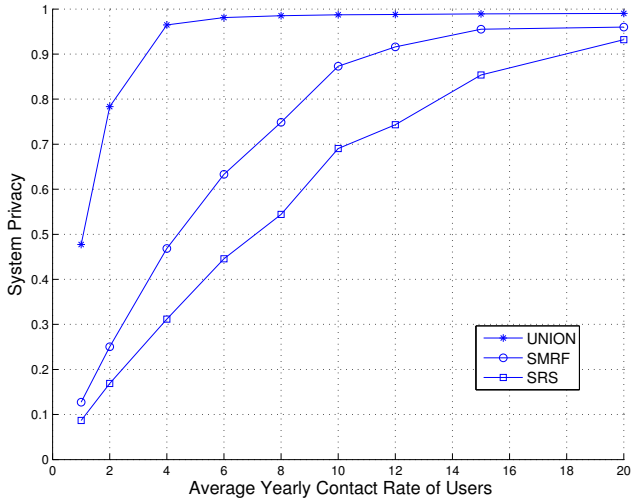
- Similarity-based minimum rating frequency (SMRF)
- Similarity-based random selection (SRS)
- Union aggregation

The Union function provides the maximum privacy that can be achieved using our method. Therefore, it acts as a benchmark to evaluate the effectiveness of the similarity-based aggregation functions.

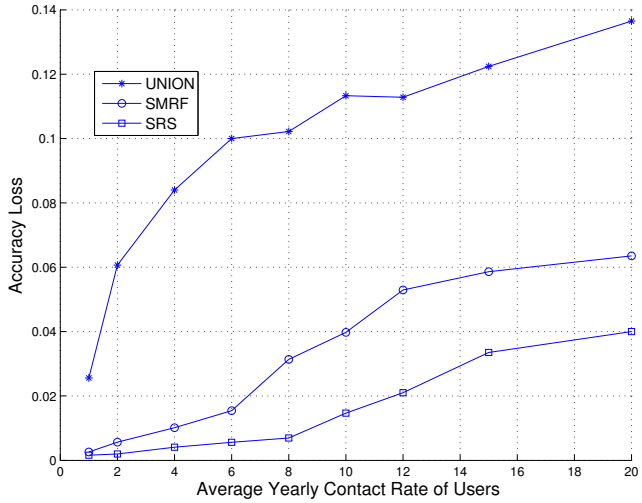
Figures 3(a) and 3(b) illustrate the privacy preservation level and accuracy loss, respectively, for different aggregation functions, calculated using Equations (11) and (13). Using these results, one can easily observe the privacy-accuracy trade-off for different mechanisms, and select the appropriate type of aggregation. The effect of contact rate on the privacy level and accuracy loss is also shown.

As depicted in Figure 3(a), using Union aggregation the system privacy level quickly approaches the maximum value of 1 as the average contact rate increases. Moreover, when this aggregation is used, the users' profiles (offline and therefore online) become more flat as time goes on (i.e., users' profiles become more similar). This leads to a considerable decrease in system accuracy, compared to other aggregation functions (Figure 3(b) shows a fast increase in accuracy loss for Union function). Contrarily, Figure 3(b) shows that the other two methods SRS and SMRF result in a slight decline in the system accuracy, in comparison with Union aggregation, for different values of contact rate.

The efficiency of our mechanism becomes more visible when we notice that a small number of contacts per user substantially increases the privacy level and imposes minor costs in terms of accuracy. For instance, notice that using the SMRF (SRS) aggregation function, number of 6 contacts per year per user results in a 0.64 (0.48) degree increment in system privacy with an accuracy loss of less than 2% (1%).



(a) Privacy gain for different users' contact rate



(b) Accuracy loss for different users' contact rate

Figure 3: Performance of different aggregation functions with respect to the achieved privacy and drop in system accuracy.

We observe that although the SRS method causes a smaller accuracy loss, the major increase in system privacy for SMRF distinguishes it as a good alternative mechanism. This is due to the fact that sorting the aggregated candidates by minimum rating frequency makes the users more indistinguishable for the server. However, SRS is the most *practical* aggregation function, as it is not dependent on any system parameter or any kind of information from the server. Another advantage of SRS over SMRF is the stability of its accuracy loss for different contact rate values.

In general, experimental results validate our proposed mechanism in preserving privacy in a distributed way. The results show that *similarity-based aggregation* achieves decent results in terms of higher system privacy with a negligible effect on accuracy loss for small average contact rates. When users have information about the rating frequency of items, giving the items with minimum rating frequency increases the privacy. This is because it increases the number of users who have rated sensitive items of a user (i.e., the items that help identifying the user). Yet, if such information is not available for users, the SRS aggregation can be used, which provides a bit less privacy but imposes a lower accuracy loss. SRS is also more stable (in terms of the accuracy loss) for different contact rate values. The key factor of similarity-based aggregation functions is that they preserve the similarity between users almost unchanged, compared to the case where there is no privacy protection mechanism in use.

6. RELATED WORK

Several techniques have been proposed to preserve the privacy of users in recommender systems. Perturbing users' ratings, using cryptographic tools such as homomorphic cryptography, and storing users' profiles in a distributed manner are the main categories for privacy preservation in collaborative filtering systems.

Polat and Du [14] propose a randomized perturbation technique to preserve privacy in collaborative filtering. Users' ratings are modified by adding random noise to them in order to prevent the central server from deriving the users' actual ratings. The challenge is to find a perturbation algorithm that imposes the smallest error on the recommendation process. The users enjoy a high level of privacy if the server is not able to estimate the actual ratings they assigned to the items. However, the items rated by the users are revealed in the proposed technique, regardless of the perturbation level. This is despite the fact that keeping the connection between users and items is more crucial (in order to preserve users' privacy) than disguising the ratings assigned to those connections. Revealing the places visited by the users to the server enables it to track users over space and time, whether they liked those places or not.

Using homomorphic cryptography in the public server in order to hide the operations of the recommender system is proposed by Canny [6, 7]. In the proposed method, users create communities and each user searches for recommendations from the most appropriate community with the hope of receiving more valuable recommendations, rather than asking from those who have similar profiles. Each community of users can compute a public aggregate of their profiles, which does not expose the individuals' profiles. Homomorphic cryptography allows the users to hide the aggregation operation from the server, although it is performed by the server itself. Participation of users in the distributed system to provide privacy for others was assumed to happen in this work, which might not be the case in reality. Moreover, the implementation of such a cryptographic scheme, especially its required key management, is difficult to achieve, considering the status of the current usage of cryptographic systems in the Internet. Similar ideas are proposed in [3] where homomorphic cryptography is used to hide similarity measurement from server's eyes.

Storing users' profiles on their own side and running the recommender system in a distributed manner, without relying on any server, is another option. Miller *et al.* [11] propose transmitting only the similarity measures over the network and keeping users' profiles secret on their side to preserve their privacy. Berkovsky *et al.* [4] propose a distributed P2P system to avoid storing users' profiles on a single server. Although these methods eliminate the main source of threat against users' privacy, they need high cooperation among users to generate meaningful recommen-

dations. Every user pays the price of using this method, regardless of his interest in protecting his privacy.

Lathia *et al.* [9] propose a concordance measure to estimate the similarity between two users in a distributed system without revealing their actual profiles to each other. A randomly generated temporal profile is shared between two users, and both of them compute the number of concordant, discordant and tied pairs of ratings between their own profile and the temporal profile. Exchanging the results, they are able to estimate the similarity between their profiles. Hence, they keep the items they have rated as well as the rating values private. In this method, users need to reveal their profiles for generating recommendations. Therefore, this method provides privacy only for measuring similarity, not for a collaborative filtering system as a whole. Two users who do not trust each other, in our proposed method, can use Lathia's method to estimate their similarity.

Finally, the concept of approximate graph matching – from which the idea of privacy metric in our work was inspired – has been widely studied in literature. The subject of structural graph matching has been investigated more specifically in pattern recognition where the goal is to match a data graph to a model graph based on nodes and edges attributes of the graphs. Myers *et al.* [12] introduce the *edit distance* measure as a metric for the structural difference of two graphs, defined as the weighted sum of the costs of edit operations to match the graphs. Also, the use of edge-consistency for computing correspondence errors in matching two graphs has been widely used, e.g., in the recent work of Luo and Hancock [10] in which they developed a likelihood function for graph matching.

7. CONCLUSION AND FUTURE WORK

In this work, we proposed a novel method for privacy preservation in collaborative filtering recommendation systems. We addressed the problem of protecting the users' privacy in the presence of an untrusted central server, where the server has access to users' profiles. To avoid privacy violation, we proposed a mechanism where users store locally an offline profile on their own side, hidden from the server, and an online profile on the server from which the server generates the recommendations. The online profiles of different users are frequently synchronized with their offline versions in an independent and distributed way. Using a graph theoretic approach, we developed a model where each user arbitrarily contacts other users over time, and modifies his own offline profile through a process known as aggregation. To evaluate the privacy of the system, we applied our model to the Netflix prize data set to investigate the privacy-accuracy tradeoff for different aggregation types. Through experiments, we showed that such a mechanism can lead to a high level of privacy through a proper choice of aggregation functions, while having a marginal negative effect on the accuracy of the recommendation system. The results illustrate that similarity-based aggregation functions, where users receive items from other users proportional to the similarity between them, yield a considerable privacy level at a very low accuracy loss.

As future work, our mechanism can be implemented in a realistic setting in which users contact each other based on their friendship (e.g., in social networks) or their physical vicinity (e.g., using wireless peer-to-peer communication). In such a setting, various practical issues such as the effect of the privacy preserving mechanism on the overhead of users' profiles and their maintenance, and the acceptability of the system in a real world scenario can be investigated. Moreover, the robustness of the algorithm to sophisticated adversarial attacks and its relation to the proposed metric are worth studying.

Acknowledgments

We would like to thank Marcin Poturalski and also the anonymous reviewers for their helpful feedback on earlier versions of this work. Special thanks go to Antoine Parisod who implemented the simulations.

8. REFERENCES

[1] Netflix prize, http://www.netflixprize.com.
[2] Rummble, http://www.rummble.com.
[3] W. Ahmad and A. Khokhar. An architecture for privacy preserving collaborative filtering on web portals. In *International Symposium on Information Assurance and Security (IAS)*, Aug. 2007.
[4] S. Berkovsky, Y. Eytani, T. Kuflik, and F. Ricci. Enhancing privacy and preserving accuracy of a distributed collaborative filtering. In *Proceedings of ACM RecSys*, 2007.
[5] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. 1998.
[6] J. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, 2002.
[7] J. Canny. Collaborative filtering with privacy via factor analysis. In *ACM SIGIR*, 2002.
[8] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *ACM SIGIR*, 1999.
[9] N. Lathia, S. Hailes, and L. Capra. Private distributed collaborative filtering using estimated concordance measures. In *Proceedings of ACM RecSys*, 2007.
[10] B. Luo and E. R. Hancock. Structural graph matching using the em algorithm and singular value decomposition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(10), 2001.
[11] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3), 2004.
[12] R. Myers, R. C. Wilson, and E. R. Hancock. Bayesian graph edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(6), 2000.
[13] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, 2008.
[14] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proceedings of IEEE ICDM*, 2003.
[15] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, 1994.
[16] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.